# Local Distributed Control of Linear Systems Despite Byzantine Faults *

Young Man Kim** Anish Arora Vinod Krishnan Kulathumani

Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210, USA
{kimyoun, anish, vinodkri}@cis.ohio-state.edu

**Abstract.** Sensor-actuator network components tend to be unreliable, especially when they are low cost or deployed in unpredictable environments. These components may even exhibit byzantine behavior in which case their effect on the underlying systems can be severe. In this paper, we focus our attention on applications of sensor networks in control of linear systems and show how to deal with byzantine faults of components. Based on the well known Lyapunov formulation of stability, we identify two types of fault-tolerant control schemes for linear systems: masking and self-stabilizing. Then, given a local control scheme which assumes that there are no byzantine faults, we develop a masking version and three increasingly efficient self-stabilizing versions that tolerate byzantine faults. We demonstrate our methodology using a beam vibration control application as a case study.

* This work was partially sponsored by DARPA contract OSU-RF #F33615-01-C-1901, NSF grant NSF-CCR-9972368, an Ameritech Faculty Fellowship, and two grants from Microsoft Research.

** On leave from the Department of Computer Science, Kookmin University, Seoul, South Korea

## 1 Introduction

A new class of distributed control applications is emerging in the field of sensor-actuator networks. Fueled in part by recent advances in micro-electromechanics and communication technologies, these applications deal with physical environments where control demands scalability in spatial, temporal, and/or resource dimensions. Scalability in turn motivates the design of control that is distributed —and often local.

By way of an example of a distributed control application, consider the control of acoustic vibrations in structures tessellated with a sensor-actuator network. Sensors enable measurement of local velocity and actuators allow application of compensating force. The sensed values are available locally as well as globally (via the network), for use by controllers in determining the actuator control signals. Mode control in this case necessitates some high frequency operations (in the kHz range) that preclude control schemes based on a global snapshot of sensor values. In other words, mode control involves distributed schemes where data from each sensor is made available only to nodes within some locality of that sensor, while still ensuring stability globally.

The distributed nature of these new control applications demands information processing services that are supported by the network. On one hand, these underlying services offer abstractions that simplify the design of the control task. Examples include controller group synchronization, communication, (re)parameterization, reconfiguration, fault detection, etc. On the other hand, these underlying services themselves introduce vulnerabilities that complicate the design of the control task. Unpredictable delays, unreliable components, insecure or compromised components, and erroneous data affect the control, potentially in severe and unanticipated ways. Early experiences in the co-design of distributed control applications and underlying services reveal not only software engineering issues —e.g., control designers expect service designers to specify highest possible service qualities whereas the latter expect the former to specify lowest service qualities that suffice for control purposes— but more importantly a lack in techniques for verifying/designing distributed control based on vulnerable network-supported services.

**Problem statement** The vulnerability of network services to faults and intruders lead us to considering the problem of designing distributed control. As a first step in this direction, we consider a simpler version of the problem:

> *Assuming that a bounded number of network nodes can exhibit incorrect (and potentially arbitrary) behavior, how can distributed control be designed to be provably stable?*

In other words, we assume that at most a bounded number of network nodes are subject to byzantine faults [1]. And since we are considering vulnerabilities of the underlying services themselves, we do not assume that there is an underlying network service for byzantine fault detection. It is therefore worth emphasizing that our approach is in contrast to several recent works that deal with faults in control systems via fault detection and subsequent isolation [2–6]. Detection

based approaches also include those which use switched linear systems [7, 8]. Switched systems are hybrid dynamical systems that consist of multiple subsystems controlled by switching laws. The switching laws are governed by the fault detectors. We eschew the detection based approach since if the detection is performed on a Byzantine node, it may lead to incorrect controller reconfiguration. Systems whose parameters are uncertain and time varying are often modeled as jump linear systems [9], when transition among the parameters is assumed to follow a given model such as Markovian. However, in this paper we deal with faults introduced by network nodes. The system parameters are assumed to be constant.

Specifically, in this paper, we consider distributed control [10–15] in the context of *linear systems* and present a series of four methods for control design that deals with Byzantine components.

**Organization of the paper** In Section 2, we recall a characterization of linear systems with control inputs and a stability condition for such systems in terms of its Lyapunov function. We then design two local control schemes that generate actuator control signals based on its local sensor values, assuming that all local controllers are fault-free. We also illustrate the effect of faults on linear systems running these schemes. In Section 3, we define two types of control schemes that deal with faults: masking and self-stabilizing. We formulate a theory for fault-tolerant control and develop four different methods for design of Byzantine fault-tolerant control. First is a *static* method for designing fault-masking control. Second is a *dynamic* method that achieves self-stabilizing control with potentially lesser energy cost than the first scheme. Third is an *adaptive dynamic* method, which with access to extra sensor state information, achieves faster convergence than the dynamic method. Fourth is a *dynamic balancer* method, which with added access to actuator state information, converges even better than the adaptive dynamic method. Each of these four methods is demonstrated in the context of a running example, a toy version of the vibration control application described above where the structure in question is a beam and the bound on the number of Byzantine nodes is just 1. In section 4, we present a comparison among the four schemes based on the energy spent in control. Lastly, in Section 5, we present conclusions and goals for future research.

## 2 Local Control Schemes for Linear Systems

In this section, we recall the description of a continuous time linear system with control inputs and with no external continuous force being applied to it, and characterize its asymptotic stability in terms of a Lyapunov function. However, the control vector for a continuous time linear system is often computed at discrete intervals of time. We then show that when the sampling frequency is high, the asymptotic stability conditions for the discrete time linear system is the same as that for continuous time systems. We then present two local control schemes for linear systems [16, 13–15] and illustrate the effect of faults on linear systems running those schemes.

### 2.1 Asymptotic Stability Condition for Continuous Time Linear Systems

A linear system with control input is represented in the standard state space format as

$$\dot{x} = Ax + Bu \tag{1}$$

where $x$ is an $n$-dimensional state vector $[x_1, x_2, \cdots, x_n]^T$, $u$ is an $m$-dimensional actuator control vector $[u_1, u_2, \cdots, u_m]$, and $A$ and $B$ are $n \times n$ and $n \times m$ matrices, respectively.

The system state vector is observed by the sensor vector $y = [y_1, y_2, \cdots, y_p]$ as

$$y = Cx \tag{2}$$

where matrix $C$ is $p \times n$.

The control scheme for the above linear system is realized in the form of an equation relating the control vector $u$ and the sensor vector $y$ as

$$u = Gy \tag{3}$$

where $G$ is an $m \times p$ gain matrix.

By inserting Eqs. 2 & 3 into Eq. 1, we get another form of linear control system, represented as

$$\dot{x} = Ax + BGCx = (A + BGC)x \tag{4}$$

A well-known asymptotic stability condition for the linear system denoted in Eq. 4 is

$$Re(\lambda_i) < 0, i = 1, \cdots, n \tag{5}$$

where $\lambda_i$ is $i$-th eigenvalue of matrix $(A + BGC)$ and function $Re(a)$ is the real part of the complex variable $a$. Note that if the real part of one or more of these eigenvalues becomes zero, the system remains stable but not asymptotically stable. However, if the real part of any of the eigenvalues becomes greater than zero, the system becomes unstable [17].

Matrices $B$ and $C$ are strongly dependent on the locations of the sensors and actuators. The gain matrix $G$ is determined by several alternative control design schemes such as pole allocation, optimal control design, Lyapunov etc., in such a way that the matrix $(A + BGC)$ makes the system asymptotically stable.

Now, let us define function $V$ as

$$V = x^T M x \tag{6}$$

where $M$ is a symmetric, positive definite $n \times n$ matrix. The time-derivative of $V$, $\dot{V}$, is derived using Eqs. 1 & 4 as

$$\begin{aligned}
\dot{V} &= \dot{x}^T M x + x^T M \dot{x} && (7)\\
&= x^T((A + BGC)^T M + M(A + BGC))x \qquad \text{, from } Eq.\ 4 && (8)\\
&= -x^T U x && (9)\\
&= x^T(A^T M + MA)x + u^T B^T M x + x^T M B u && (10)\\
&= x^T(A^T M + MA)x + 2x^T M B u && (11)
\end{aligned}$$

where matrix $U$ is a positive definite, symmetric matrix.

It is well-known that the linear system denoted in Eq. 4 is asymptotically stable iff there exist positive definite, symmetric matrices $M$ and $U$ in Eq. 9 [18, 19]. Notice also from Eq. 9 that irrespective of the value of the state variable $x$ in its entire state space, $\dot{V}$ is always negative. Hence, $V$ is a Lyapunov function.

If the value of the Lyapunov derivative becomes strictly positive for any value of $x$ due to the occurrence of faults, which consequently lead to faulty control signals, the system becomes unstable. Let the *stability margin* $S(x)$ of the system in the state $x$ denote the margin of the system from the unstable region.

$$S(x) = x^T(A^TM + MA)x + 2x^TMBu \tag{12}$$

Thus, S(x) is equal to $\dot{V}$. Note that under normal fault free operation, $S(x) = -x^tUx$ and the stability margin is always negative.

### 2.2 Asymptotic Stability Condition for Discrete Time Linear Systems

The control vector for a linear system is often computed at discrete time intervals. Suppose that the period of each interval is $T$. We now show that when the sampling frequency is high, the rate of change of the Lyapunov function stays the same as in Eq. 11.

The computed control vector $u(kT)$ at instant $kT, k >= 0$, is fed-back into the system during the interval $[kT, (k+1)T)$. Note that this control vector does not change its value during that period. Then the discrete time system can be modeled by the following state equations [20]:

$$x((k+1)T) = e^{AT}x(kT) + BTu(kT) \tag{13}$$

$$= (I + (AT) + \frac{1}{2!}(AT)^2 + .....)x(kT) + BTu(kT) \tag{14}$$

If the period $T$ is small enough, it is reasonable to omit the higher powers of $AT$. Hence the equation gets the simpler form:

$$x((k+1)T) = (I + (AT))x(kT) + BTu(kT) \tag{15}$$

The Lyapunov function for the discrete time linear system is modeled as:

$$V(kT) = x^T(kT)Mx(kT) \tag{16}$$

where M is a symmetric positive definite $n \times n$ matrix.

The rate of change of the Lyapunov function is derived as follows:

$$\Delta V(kT) = (V(k+1)T - V(kT))/T \tag{17}$$

$$\simeq x^T(kT)((A + BGC)^TM + M(A + BGC))x(kT) \tag{18}$$

where the second order terms attached with $T^2$ are ignored.

Thus we see that the rate of change of the Lyapunov function is equal to the Lyapunov derivative obtained by Eq. 11.

*From here on for simplicity of notation, we denote $V(kT)$ as $V$, $x(kT)$ as $x$ and so on for each state variable. Also, we denote $\Delta V(kT)$ as $\dot{V}$.*

In the following subsection, we recall two local control schemes for linear systems and employ the Lyapunov function $V$, to analyze their stability margins.

### 2.3 Local Control Schemes Assuming No Faults

A distributed control system consists of a dynamic physical plant and multiple nodes that are connected by a wired or wireless network. Each node consists of a sensor, an actuator, a processor, and network services. It measures the plant state with its sensor, exchanges control information with the other nodes using its network services, computes the control signal according to its control logic, and applies the signal to the plant through its actuator.

In this section, we recall two local control schemes for linear systems and employ the Lyapunov function $V$, to analyze their stability margins. In the first scheme, the control vector generated is proportional to the sensor vector. This is called as the linear local control scheme [16, 13–15]. In the second scheme, the control inputs are binary valued. This is called as the on-off local control scheme. Restating Eq. 11,

$$\dot{V} = x^T(A^TM + MA)x + 2x^TMBu.$$

If matrices $A$, $M$, $C$ and $B$ satisfy the following conditions, then the system is controlled locally, i.e., no control information is exchanged between nodes.

$$C^TC = A^TM + MA \tag{19}$$

$$C = B^TM \tag{20}$$

Based on these two conditions, the Lyapunov derivative becomes

$$\begin{aligned}\dot{V} &= x^T(A^TM + MA)x + 2x^TMBu \\ &= x^T(C^TC)x + 2x^T((B^TM)^T)u \\ &= x^T(C^TC)x + 2x^T(C)^Tu \\ &= (Cx)^TCx + 2(Cx)^Tu \\ &= y^Ty + 2y^Tu \end{aligned} \tag{21}$$

Let the system have $m$ nodes.

$$\dot{V} = \sum_{i=1}^{m}((y_i)^2 + 2y_iu_i) \tag{22}$$

Recall that the Lyapunov derivative is interpreted as the stability margin for the system. If we were to specify a constant stability margin $s$ for the system, where $s$ is a negative number, then the control force $u_i$ would have to be very large when the sensor vector $y_i$ was very small. Hence, we specify our requirement for the stability margin S(x) as a value proportional to the sensor vector:

$$S(x) = s \cdot f(y_i) \tag{23}$$

where $s$ is the stability margin constant for the system and $f(y_i)$ is a function of the sensor vector. Consider the following linear local control scheme.

**Linear Local Control Scheme**

*Input:* local sensor data $y_i$ ; $s$
*Output:* local control signal $u_i$
*Procedure:*
**while** true **do**
$u_i = \frac{1}{2}(s-1)y_i$
**od**

The above scheme generates $u$ such that

$$\begin{aligned} \dot{V} &= \sum_{i=1}^{m}((y_i)^2 + 2y_i u_i) \\ &= s \cdot \sum_{i=1}^{m}(y_i)^2 \end{aligned} \tag{24}$$

Thus, the stability margin for this system is $s \cdot \sum_{i=1}^{m}(y_i)^2$. Since $u$ is derived from the local sensor data without computing the state vector, this scheme is also called direct feedback control [21, 22].

An alternative local control scheme is an on-off scheme [23], where the control inputs are binary valued. Thus, for all $i$, $u_i$ is either $+N$ or -$N$, where $N$ is a positive value such that $y_i < 2N - |s|$.

**On-Off Local Control Scheme**

*Input:* local sensor data $y_i$ , stability margin (with negative value) $s$
*Output:* local control signal $u_i$
*Procedure:*
$N = (max(|y_i|) + |s|)/2$
**while** true **do**
$u_i = -N \times sign(y_i)$
**od**

For this on-off control scheme, we have from Eq. 21:

$$\dot{V} = s \times \sum_{i=1}^{m} |y_i| \tag{25}$$

Thus the stability margin for this system is $s \times \sum_{i=1}^{m} |y_i|$.

The application of each of these schemes is demonstrated in the context of a simplified version of the beam vibration control system outlined in Section 1, which we describe next.

### 2.4 Beam Vibration Control System : A Running Case Study

Given is a uniform beam of unit length, unit mass, and unit stiffness factor, that is restricted by pins at both ends and subjected to an initial disturbance. The

beam has no dampening factor so that it may vibrate endlessly. Control force is applied to reduce the vibration. (Such a system is called an energy conserving control system.)

We assume that only two fundamental vibration modes are significant in the vibration and the remaining ones are hence ignored in our analysis.

The two fundamental vibration modes, denoted as $M_1$ and $M_2$, are derived [17] as follows:

$$M_1 : 1.4142 \sin \pi z, \quad \lambda_1 = \omega_1^2 = 97.4091 \tag{26}$$

$$M_2 : 1.4142 \sin 2\pi z, \quad \lambda_2 = \omega_2^2 = 1558.5455 \tag{27}$$

where $z \in [0.0, 1.0]$ denotes the position in the beam spatial axis and $\lambda_i$ and $\omega_i$, $i = 1, 2$, represent the eigenvalues and the frequencies of $i$-th modes, respectively.

Since each mode is governed by a second-degree differential equation, the state vector for the system contains four variables $x = [x_1, x_2, x_3, x_4]^T$. $x_1(x_2)$ and $x_3(x_4)$ denote the vertical displacement and velocity of first (second) vibration mode, respectively. Then, the system matrix $A$ in Eq. 1 is denoted as

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -97.4091 & 0 & 0 & 0 \\ 0 & -1558.5455 & 0 & 0 \end{bmatrix}$$

Since the beam is an energy conserving dynamic system, the energy function can be employed as the Lyapunov function. The energy or Lyapunov function $E$ and its derivative $\dot{E}$ are derived as follows:

$$E = \frac{1}{2} x^T M x \tag{28}$$

$$\text{where} \quad M = \begin{bmatrix} 97.4091 & 0 & 0 & 0 \\ 0 & 1558.5455 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

$$\dot{E} = x^T M B u = y^T u \tag{29}$$

Note that the equation above differs from the Lyapunov derivative Eq. 21, in that the term $y^T y$ is now missing. This is due to energy conservation, which makes $A^T M + MA$ a null matrix. Note also that the term $y^T u$ has multiplicative constant 1 instead of 2 since the definition of the energy function has a multiplicative constant $\frac{1}{2}$ instead of 1.

As mentioned previously, the sensor matrix $C$ and the actuator matrix $B$ are dependent on the location of the nodes. In what follows, let us consider a distributed control system for the beam that consists of three nodes, located at the following positions: $z = 0.221, 0.236, 0.5$. In this particular configuration, from Eqs. 26 & 27, the matrices $B$ and $C$ are

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.9049 & 0.9551 & 1.4142 \\ 1.3908 & 1.4087 & 0 \end{bmatrix}$$

In the above matrices, each element is computed from the mode equations, Eq. 26-27. As an example, the influence constant for the node at $z = 0.221$ in mode $M_1$ is computed using Eq. 26, i.e., $0.9049 = 1.4142 \sin \Pi * 0.221$, and the influence constant for the node at $z = 0.221$ in mode $M_2$ is computed using Eq. 27 as $1.3908 = 1.4142 \sin 2 * \Pi * 0.221$ and so on.

$$C = B^T M = B^T = \begin{bmatrix} 0 & 0 & 0.9049 & 1.3908 \\ 0 & 0 & 0.9551 & 1.4087 \\ 0 & 0 & 1.4142 & 0 \end{bmatrix}$$

The linear and on-off local control schemes for linear systems described in the previous subsection can be adapted for this energy conserving system as follows:

$$\text{Linear control} : u_i = g_i \cdot y_i \tag{30}$$

$$\text{On-off control} : u_i = l_i \cdot sign(y_i) \tag{31}$$

where $g_i$ is a negative number, the local gain factor and $l_i$ is a negative number, the local on-off level. Note that greater the gain $g_i$, greater the stability margin for linear control. For the on-off control scheme, the level $l_i$ has to be greater than the maximum of $y_i$. (See section 2.2) Again greater this level, greater the stability margin. The effect of different values of gain and on-off level, on the linear system are illustrated in the simulation below.

*Simulation* Simulations of all the control schemes discussed in this paper have been done in Matlab using Simulink. The graphs in Figs. 1 and 2 show the energy in the beam decreasing with time and eventually reaching zero, for the linear control scheme and the on-off control scheme. The three sensor-actuator nodes are placed at $z = 0.221, 0.236, 0.5$. It is observed that the time to stabilize the beam is inversely proportional to the gain and the on-off level. Note that in each of our simulations, when the energy in the beam falls to a reasonably low threshold, e.g. 0.5% of the original value, the system is said to have stabilized.

### 2.5 Effect of Faults on the Beam Vibration System

In this section, we illustrate the effect of a faulty node on the quality of control that is achieved. In one experiment, the node at location $z = 0.5$ was made to generate random values. In another experiment, that node was made to generate the maximum force in the opposite direction to what was required. The graph in Fig. 3 shows the energy in the beam with respect to time, in both these experiments. The faults are observed to be intolerable.

The rest of the paper outlines a fault model for such a linear control system and describes control schemes that are (masking or self-stabilizing) fault-tolerant.

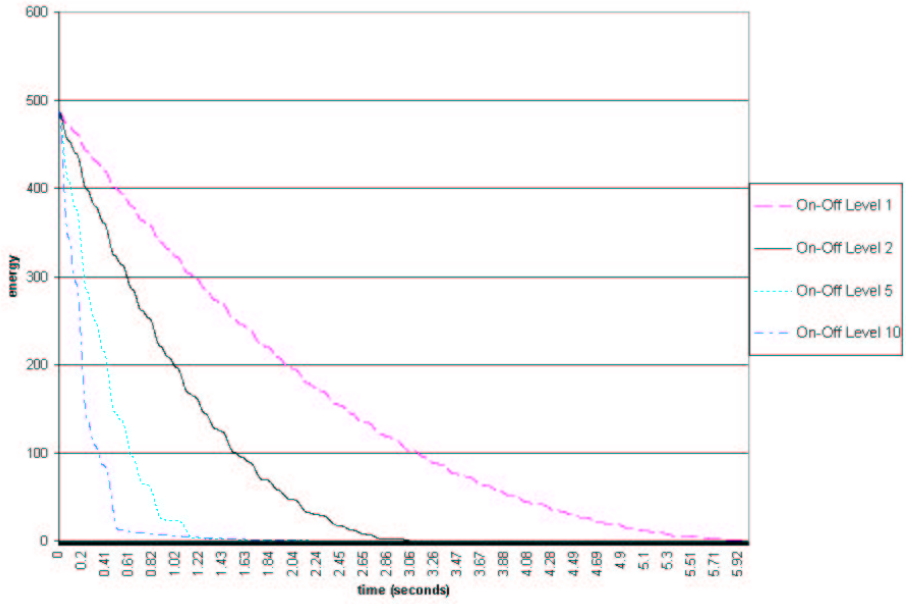


**Fig. 1.** Comparison of On-Off Control for Different On-Off Levels (energy vs time)

## 3 Fault-Tolerant Control Schemes

In this section, we develop a little theory of fault-tolerant linear systems, present a fault model for a linear control system and devise four fault-tolerant control schemes.

### 3.1 Fault-Tolerant Linear Systems

Given a faulty control vector $u^f$, let the *fault index set* of $u^f$ be the set of indices of faulty control signals, $I(u^f, u) = \{i : 1 \leq i \leq n, u_i^f \neq u_i\}$. Let the *control failure levels* $F(i, u^f, u, x)$, for $i \in I(u^f, u)$, and $F(u^f, u, x)$ measure the impact of the faulty control signal $u_i^f$ and the faulty control vector $u^f$, respectively, on the Lyapunov derivative in Eq. 9.

$$F(i, u^f, u, x) = 2(x^T MB)_i(u_i^f - u_i) \tag{32}$$

$$F(u^f, u, x) = \sum_{i \in I(u^f, u)} F(i, u^f, u, x) \tag{33}$$

Now, faulty control signals may or may not affect system stability. If the control failure level is positive (negative), system stability deteriorates (enhances) the stability. Under the influence of the faulty control vector $u^f$, the stability margin $S(u^f, x)$ is

$$S(u^f, x) = x^T(A^T M + MA)x + 2x^T MBu^f = F(u^f, u, x) - x^T Ux. \tag{34}$$

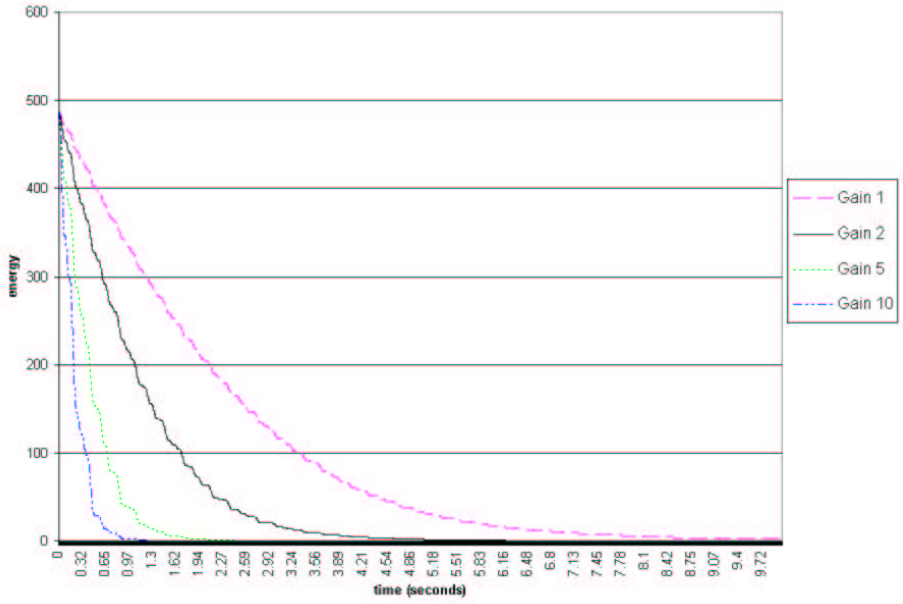


**Fig. 2.** Effect of Gain on Linear Control (energy vs time)

If $S(u^f, x)$ is always negative, the linear system is asymptotically stable even in the presence of faults, and is fault-tolerant. If it is not always negative, fault-tolerance requires that the values of some normal control signals be modified so as to shift the stability margin into the negative region. Suppose that a fault-tolerant control vector $u^t$ is generated by one such modification scheme. Let the *fault-tolerant index set* for control vectors $u^t$, $u^f$ and $u$ be the set of indices of fault-compensating control signals, $J(u^t, u^f, u) = \{i : 1 \leq i \leq n, u_i^t \neq u_i^f\}$. Let the *control stabilization levels* $P(i, u^t, u^f, x)$, for $i \in J(u^t, u^f, u)$, and $P(u^t, u^f, x)$ measure the impact of fault-compensation signal $u_i^t$ and fault-compensation control vector $u^t$ respectively on the Lyapunov derivative in Eq. 9.

$$P(i, u^t, u^f, x) = 2(x^T MB)_i(u_i^t - u_i^f) \tag{35}$$

$$P(u^t, u^f, x) = \sum_{i \in J(u^t, u^f)} P(i, u^t, u^f, x) \tag{36}$$

The new *fault-tolerant stability margin* for the fault-tolerant control input $u^t$ is

$$S(u^t, u^f, x) = x^T(A^T M + MA)x + 2x^T MBu^t \tag{37}$$

$$= P(u^t, u^f, x) + F(u^f, u, x) - x^T Ux \tag{38}$$

We may now identify two types of fault-tolerant control.

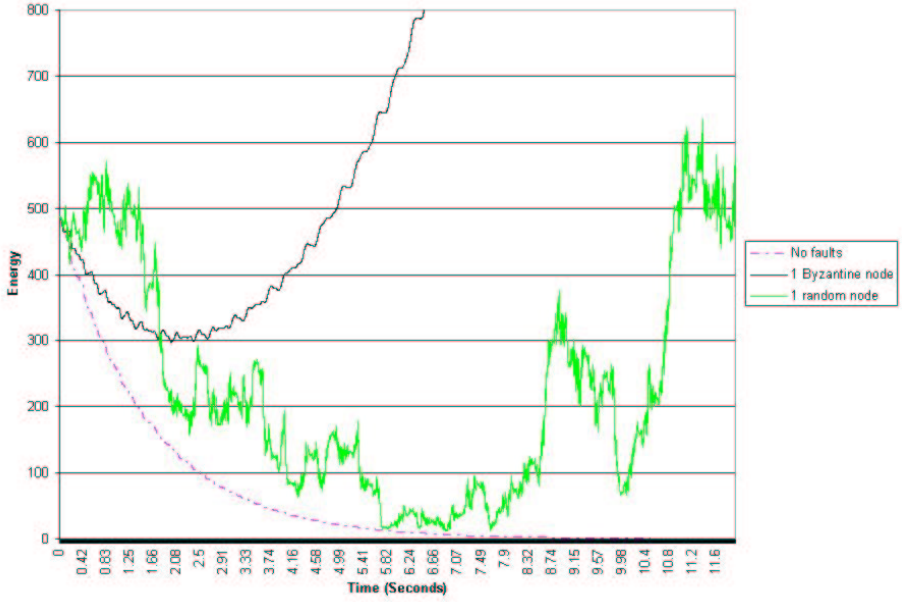


**Fig. 3.** Effect of Faults on Stability of Beam Vibration Control (energy vs time)

*Definition. (Masking)* If a linear system $L$ always satisfies the condition $S(u^f, x) < 0$ over the entire domain of $u^f$ and $x$, except at the equilibrium point $x = 0$, then $L$ is masking fault-tolerant.

*Definition. (Self-Stabilizing)* If linear system L and its control vector $u_t$ eventually always satisfy the condition $S(u^t, u^f, x) < 0$ over the entire domain of $u^f$ and $x$, except at the equilibrium point $x = 0$, then L is self-stabilizing fault-tolerant.

In other words, a linear system is fault-tolerant if it is either masking (i.e. always its stability margin is negative) or self-stabilizing (iff eventually always its stability margin is negative).

### 3.2 Fault Model

The system contains at most $k$ Byzantine nodes, for a given bound $k$. A byzantine node may behave in an arbitrary manner, in its sensing, its processing, its actuation, and its communications. In particular, it may apply an arbitrary control force (chosen from the bounded domain of the actuator) and can make the system unstable.

The amount of information that is available in the system affects the ability of a control scheme to tolerate byzantine faults. On one hand, if the control failure level in the presence of byzantine faults can be measured, fault-tolerance is readily achieved. On the other hand, such failure information is often not available. In the following subsections, we propose four fault-tolerant control

schemes that vary in the amount of failure information that is available to them. The application of each of these schemes is demonstrated in the context of a simplified version of the beam vibration control system described in Section 2.

### 3.3 Static Fault-Tolerant Scheme

One approach to designing a fault-tolerant control scheme is to pessimistically assume that all byzantine nodes generate the worst possible actuator control signal (so as to drive the Lyapunov derivative farthest into the positive domain), and to then choose a configuration of nodes such that the system remains asymptotically stable over the entire domain of the state space $x$. We call this approach the *static fault-tolerant scheme.*

The static scheme performs a node configuration search starting with $k+1$ nodes, where $k$ is the bound on the number of byzantine nodes. Given the number of nodes, their locations, and assuming the worst possible signals from the byzantine nodes, it expresses the Lyapunov derivative as a nonlinear equation. It then partitions the state space into a set of subdomains so that, in each subdomain, the equation becomes linear. If in all the subdomains the Lyapunov derivative remains negative, that configuration constitutes a masking fault-tolerant control system. If no such configuration exist, the number of nodes are increased and the search is continued.

The static fault-tolerant scheme is described below, with control assumed to be on-off. Formally, the static fault-tolerant scheme is:

**Static Fault-Tolerant Control Scheme**

*Input:* On-off local control scheme ($H$), $k$
*Output:* node configuration (the number of nodes and their locations)
*Procedure:*
set $numberofnodes = k - 1$
**while** true **do**
  increment $numberofnodes$ by one
  **for** each possible locations for the current set of nodes
    compute sensor and actuator matrices $C$ and $B$
    partition the total state space domain into a set of linear subdomains
      $\{P_1, P_2, \cdots, P_l\}$
    **for** each subdomain $P_i$, $1 \leq i \leq l$
      **for** each subset of k nodes in the current configuration
        assume the nodes in the subset are byzantine
        compute the maximum faulty control margin $S(u_f, x)$
        **if** there exists a location $x \in P_i$ (other than $x = 0$) s.t. $S(u_f, x) >= 0$
        **then** continue
        **else** return the current configuration data
**od**

**Application of the Scheme in Beam Vibration Control** To simplify finding a masking fault-tolerant scheme, we restrict the $n$-node configuration in the beam to satisfy the following pattern:

$$z_i = 0.25 + 0.014(i - n - 0.5), 1 \leq i \leq n, n \geq 2 \tag{39}$$

In other words, the $n$ nodes are distributed around the central location 0.25 with uniform distance 0.014. The following theorem shows that this configuration makes the beam vibration distributed control system masking fault-tolerant to one byzantine node.

**Theorem 1.** *The five node on-off local control scheme for beam vibration that satisfies Eq.39 is masking fault-tolerant to one byzantine fault.*

*Proof.* Let $L$ be the level for on-off control at each node. Note that, with $n = 5$, $z_i$ is equal to $0.222, 0.236, 0.25, 0.264, 0.278$. Then, from Eqs. 26 & 27, the influence matrix $B$ and the sensitivity matrix $C$ are

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.9049 & 0.9551 & 1.0 & 1.043 & 1.0868 \\ 1.3908 & 1.4087 & 1.4142 & 1.4087 & 1.3908 \end{bmatrix}$$

$$C = B^T = \begin{bmatrix} 0 & 0 & 0.9049 & 1.3908 \\ 0 & 0 & 0.9551 & 1.4087 \\ 0 & 0 & 1.0 & 1.4142 \\ 0 & 0 & 1.043 & 1.4087 \\ 0 & 0 & 1.0868 & 1.3908 \end{bmatrix}$$

The Lyapunov derivative for the five node beam vibration system is:

$$\begin{aligned} \dot{E} &= y^T u = (Cx)^T(-L * sign(y_i) \\ &= -L \times |0.9049x_3 + 1.3908x_4| - L \times |0.9551x_3 + 1.4087x_4| \\ &\quad -L \times |1.0x_3 + 1.4142x_4| - L \times |1.043x_3 + 1.4087x_4| \\ &\quad -L \times |1.0868x3 + 1.3908x4| \end{aligned}$$

When there are no faulty nodes, control is applied in the direction opposite to the sensed force. Hence the energy derivative is always negative. But now consider the case when one of the nodes is byzantine. It always applies a force of level $L$ in the wrong direction.

Since the above equation is nonlinear, first it is partitioned into a set of linear equations. The five linear equations, obtained by equating the five absolute terms in the Lyapunov derivative to zero, partition the system state space into ten subdomains.

For example, suppose that subdomain $R_1$ represents the region in which all inside equations in the absolute terms are positive. Then, the Lyapunov derivative in $R_1$ becomes $s(4.9898x_3 + 7.0276x_4)$. Suppose that the node at $z = 0.264$ is the byzantine node, the fourth term in the Lyapunov derivative is set with

the opposite sign and the corresponding Lyapunov derivative in subdomain $R_1$ becomes $s(2.9038x_3 + 4.2102x_4)$. This is less than zero, except at the equilibrium point when it is zero. Using the same approach, the Lyapunov derivative can be shown to be less than zero in all subdomains, irrespective of whichever node goes faulty. □

**Corollary 1.** *Every five node on-off local control scheme for beam vibration that satisfies Eq.40 is masking fault-tolerant to one byzantine fault.*

$$z_i = 0.25 + d(i - k - 0.5), \quad 1 \leq i \leq k, \ k \leq 2, \ d \in [0, 0.67] \tag{40}$$

*Proof.* The proof is similar to that for Theorem 1 and is skipped here. □

*Simulation* In Fig. 4, we show the energy in the beam plotted against time (in seconds) for the static scheme. Each node generates either $+5$ or $-5$ units of force, in the opposite direction of the sensed force. Three cases are considered: no faults in the system, one node generating the force in a random direction and one byzantine node which generates the force always in the wrong direction. On comparing with Fig. 3, we observe that with the static scheme, the energy in the beam decreases in each of the fault scenarios.

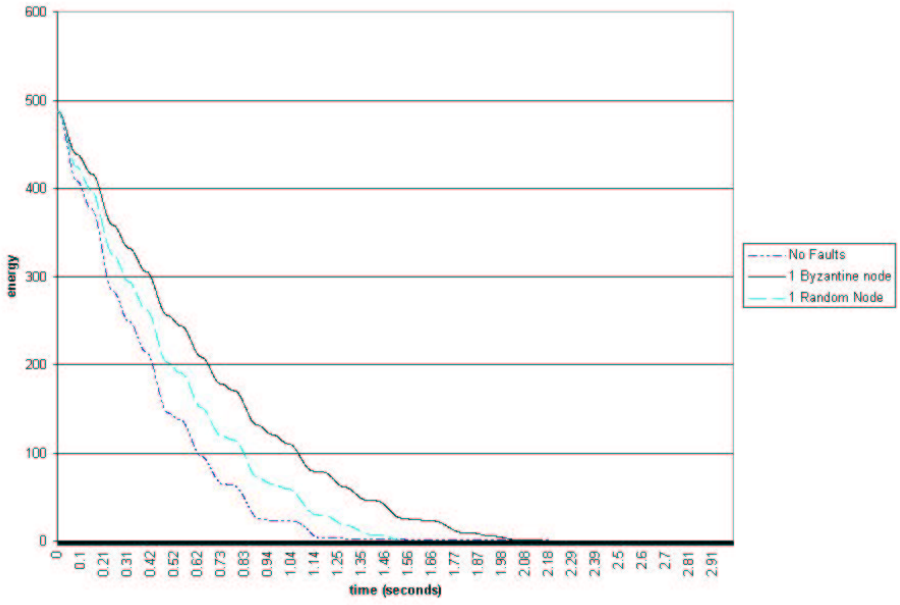


**Fig. 4.** Static Fault-Tolerant Scheme (energy vs time)

In the static scheme, irrespective of the magnitude of the fault or the magnitude of the sensed force, each node generates a constant force, resulting in huge energy spent in control. However, it is possible that the faults may not be very

severe and they can be tolerated by applying less than the maximum force all the time and/or with fewer nodes. The following linear schemes are based on this idea. A comparison of the energy spent in control in all the schemes, with and without faults, is made in Section 5.

### 3.4 Dynamic Fault-Tolerant Scheme

In this scheme, we relax the pessimistic assumption that byzantine nodes generate the worst possible control signals. The scheme increases the control force at each node linearly from a certain minimum value. Thus if the byzantine faults are not very severe, they can be tolerated with lesser energy. The dynamic scheme thus yields more energy efficient control than the static scheme.

**Dynamic Fault-Tolerant Control Scheme (at $k$-th node)**

*Input:* positive gain adjustment rate $\Delta$, time unit length $t_b$,
node configuration (output by static fault-tolerant scheme)
*Output:* local gain factor $g_k$
*Procedure:*
(Initially)
set $g_k = sy_k$ (From Eq. 30)
set $i = 1$ and $S_{-2} = S_{-1} = 0$
**while** true **do**
at the beginning of time interval $I_i$
let $S_i = E_{i-1} - E_{i-2}$
**if** $S_i < 0$ **then** $S_i = 0$
update $g_k = g_k - \Delta \cdot S_i$
increment $i$ by one
**do**

*Description* Time is partitioned into uniform intervals of length $t_b$. One choice for $t_b$ is the period of the fundamental frequency mode. Let $E_i$ be the system excitation level of the $i$-th interval, $I_i = [(i-1)t_b, it_b)$. This is equal to the maximum sensor value $y_k$, sensed during the $i$th interval at node $k$. Let the system stability level ($S_i$) at the beginning of $i$-th interval($I_i$) be the difference $E_{i-1} - E_{i-2}$. If the stability level $S_i$ is negative, then the system is stable at the beginning of $I_i$, otherwise it is unstable. Depending on the value of $S_i$, the control gain is updated to improve the stability level during time interval $I_i$. From the static scheme, we have the safety guarantee that when the control forces are maximum, the system will be asymptotically stable.

**Application of the Scheme in Beam Vibration Control** In the beam vibration example, the 2 modes have frequencies 9.87 Hz and 39.48 Hz. Hence, without control input, the local velocity profile repeats itself every $1/9.87 = 101 milliseconds$. If a control input is applied on the beam, the local velocity should decrease at least every 101ms. Hence, $t_b$ is input as $101ms$. The location of the nodes and the matrices B and C remain the same as in the static fault-tolerant scheme.

**Theorem 2.** *The five node distributed control system for beam vibration resulting from the dynamic fault-tolerant scheme is self-stabilizing fault-tolerant.*

*Proof.* We know that the 5 nodes on-off distributed control system tolerates one byzantine fault. Hence, in the dynamic system, when the gain factor becomes sufficiently high so as to get the control forces to the maximum, the system is guaranteed to stabilize. □

*Optimization* Instead of activating all 5 nodes, we can start off by activating just 2 nodes. If the local velocities do not drop after say 10 $t_b$, an additional node can be activated until at most all five are activate. Note, however, that this optimization requires a centralized controller.

*Simulation* In Fig. 5, we show the energy in the beam plotted against time (in seconds) for the dynamic scheme. In order for this analysis to be comparable with the analysis of the static scheme, the on-off level in the static scheme was set to 5 and the actuator forces in the dynamic scheme were bounded by 5. Thus, the local gain factors in the dynamic scheme can increase and bring each actuator force to at most equal the on-off level of the static scheme. We compare the performance with that of the static scheme in the following three cases:

- no faulty nodes in the system : desired control is achieved with 2 nodes in the system,
- one node that generates random forces : desired control is achieved with 3 nodes in the system, and
- one byzantine node which generates maximum force in the wrong direction : desired control is achieved with 4 nodes in the system.

### 3.5 Dynamic Adaptive Fault-Tolerant Scheme

In the previous scheme, we used the local excitation levels as a measure of system stability. This is however an approximate measure and the time interval $t_b$ required to make the gain adjustment is relatively large and can result in significant convergence time.

To calculate the stability measure exactly, some global knowledge of the system state is required. For example, the Lyapunov function can be computed if the correct system state vector is available. The expected Lyapunov derivative can then be calculated using the state vector and Eqs. 2 and 3. However, without measuring the actuator force, the real Lyapunov derivative cannot be computed. So we use the following approximation. Suppose that the Lyapunov functions $V_{i-1}$ and $V_i$ denote the value at the beginning of intervals $I_{i-1}$ and $I_i$, respectively. Then, the actual Lyapunov derivative $\hat{\dot{V}}_i$ can be approximated as $\frac{V_i - V_{i-1}}{t_b}$. Suppose that the expected Lyapunov derivative at the beginning of interval $I_i$ is $\bar{\dot{V}}$. Then, the difference between the actual and the expected Lyapunov derivative can be exactly compensated by incrementing the gain. The following scheme implements this idea (the method for computing the correct state vector from the sensing data is application specific).

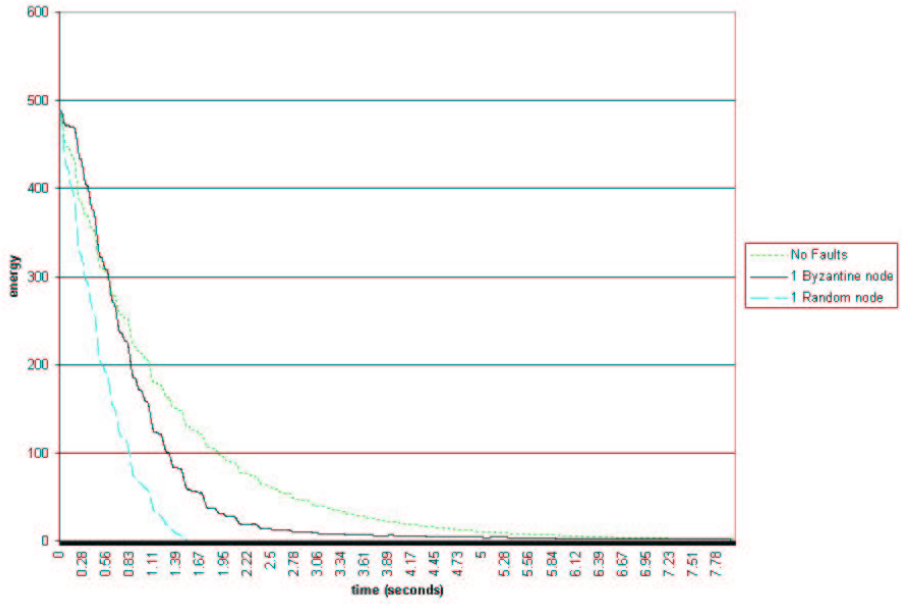


**Fig. 5.** Dynamic Fault-Tolerant Scheme (energy vs time graph)

### Dynamic Adaptive Fault-Tolerant Control Scheme

*Input:* time interval length $t_b$,
node configuration (output by static fault-tolerant scheme)
*Output:* gain factor $g_k$
*Procedure:*
(Initially)
set $g_k = sy_k$ (From Eq. 30)
set $i = 1$ and $V_0 = V_1$
**while** true **do**
at the beginning of time interval $I_i$
let $\hat{\dot{V}}_i = \frac{V_{i-1} - V_{i-2}}{t_b}$
**if** $\hat{\dot{V}}_i < \bar{\dot{V}}_i$ **then** $\hat{\dot{V}}_i = \bar{\dot{V}}_i$
additional gain $\Delta g_k$ to provide margin, $(\bar{\dot{V}}_i - \hat{\dot{V}}_i)$
increment $i$ by one
**od**

If the Lyapunov function after time $t_b$ is larger than expected, the local gain factor is increased. When the gain factor becomes sufficiently large, the control force generated will converge to that in the on-off scheme. Thus, from the asymptotic stability of the static distributed control system, it is guaranteed that even if the worst actuator input is injected by the byzantine node, the system will eventually become asymptotically stable. The time bound $t_b$ can

be made much lesser than in the dynamic fault-tolerant control scheme. Thus, the dynamic adaptive fault-tolerant scheme achieves faster convergence than the dynamic fault-tolerant scheme.

**Application of the Scheme in Beam Vibration Control** To compute the Lyapunov function, we need the state velocity vectors $x_3$ and $x_4$ for the beam. These state velocity vectors can be computed using any 2 correct local sensor vectors $y_k$. However, any of the nodes could be byzantine and hence their sensor values corrupt. The following theorem helps us determine the correct state velocity vectors for the beam.

**Theorem 3.** *Four sensors located anywhere along the beam suffice to correctly determine the state velocity vectors for the beam, given at most one byzantine node.*

*Proof.* Since the state vector comprises two velocity variables (x3 and x4), two correct sensor values are sufficient to determine the state velocity vectors. As an example, with the help of two nodes located at z=0.221 and z=0.279, the two modal velocities can be derived from the sensor data $y_1$ and $y_2$ as follows

$$\begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0.9049 & 1.3908 \\ 1.0868 & 1.3908 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{41}$$

With four sensors, we have six pairs of sensor values that can give us the state velocity vectors. Given only one byzantine node, three of those six pairings will not contain the byzantine value. □

**Dynamic Adaptive Control Scheme for Beam Vibration (at $k$-th node)**

*Input:* time interval length $t_b$,
node configuration (output by static fault-tolerant scheme)
*Output:* gain factor $g_k$
*Procedure:*
(Initially)
set $g_k = sy_k$ (From Eq. 30)
set $i = 1$ and $V_0 = V_1$
**while** true **do**
at the beginning of time interval $I_i$
let $\hat{\dot{V}}_i = \frac{V_{i-1} - V_{i-2}}{t_b}$
**if** $\hat{\dot{V}}_i < \bar{\dot{V}}_i$ **then** $\hat{\dot{V}}_i = \bar{\dot{V}}_i$
additional gain $\Delta g_k = \frac{\bar{\dot{V}}_i - \hat{\dot{V}}_i}{\sum_{i=1}^{5} y_i^2}$
update gain $g_k = (g_k + \Delta g_k)$
increment $i$ by one
**od**

**Theorem 4.** *The five node distributed control system for beam vibration resulting from the dynamic adaptive fault-tolerant scheme is self-stabilizing fault-tolerant.*

*Proof.* We know from the 5 nodes on-off distributed control system tolerates one byzantine fault. Hence, in the dynamic adaptive system, when the gain factor becomes sufficiently high so as to get the control forces to the maximum, the system is guaranteed to stabilize. □

*Optimization.* Instead of activating all 5 nodes, we can again start off by activating just 2 nodes. If the local velocities do not drop after say 10 time intervals, an additional node could be activated until all five nodes are active.

*Simulation.* In Fig. 6, we show the energy in the beam plotted against time (in seconds) for the dynamic scheme. We compare the performance with that of the earlier schemes in the following three cases:

- no faulty nodes in the system : desired control is achieved with 2 nodes in the system,
- one node that generates random forces : desired control is achieved with 3 nodes in the system, and
- one byzantine node which generates maximum force in the wrong direction : desired control is achieved with 4 nodes in the system.

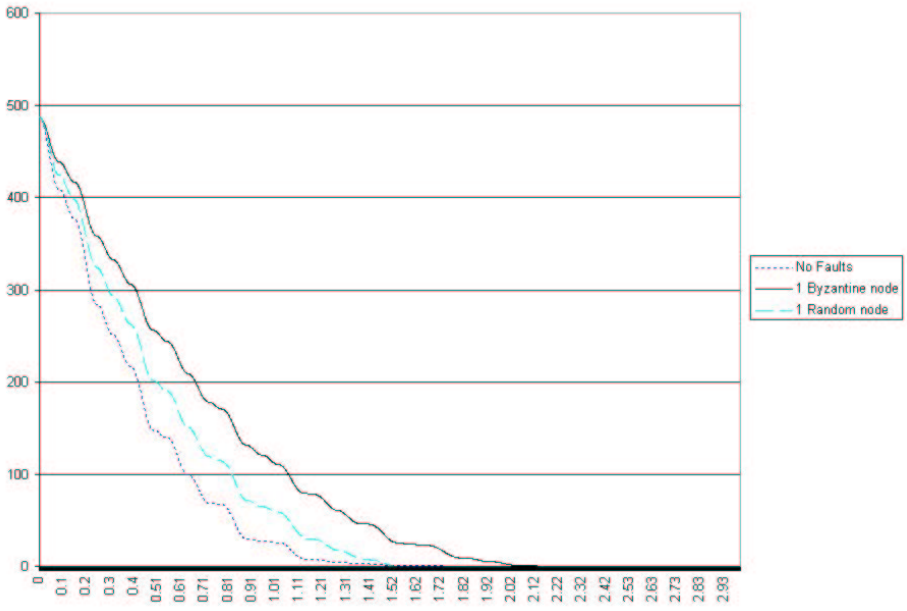


**Fig. 6.** Adaptive Dynamic Fault-Tolerant Scheme (energy vs time)

### 3.6 Dynamic Balancer Scheme

In the previous scheme, the actual Lyapunov derivative at time $t$ was approximated by using the difference between two previous Lyapunov functions. If the applied actuator forces can also be measured at each location, the exact Lyapunov derivative can be computed. The fault can then be precisely balanced out by the normal nodes. This is the idea underlying the *fault-balancer scheme.*

**Dynamic Balancer Control Scheme**

*Input:* time interval length $t_b$,
node configuration (output by static fault-tolerant scheme)
*Output:* local gain factor $g_k$
*Procedure:*
(Initially)
local gain factor, $g_k$, is set as $\frac{1}{2}(s-1)$
so that $u_k = g_k \cdot y_k = \frac{1}{2}(s-1)y_k$ (see Section 3)
set $i = 1$
**while** true **do**
at the beginning of time interval $I_i$
compute the actual Lyapunov derivative $\hat{\dot{V}}_i$
compute the expected Lyapunov derivative $\bar{\dot{V}}_i$
**if** $\hat{\dot{V}}_i < \bar{\dot{V}}_i$ **then** $\hat{\dot{V}}_i = \bar{\dot{V}}_i$
compute the additional gain $\Delta g_k$ to provide margin, $(\bar{\dot{V}}_i - \hat{\dot{V}}_i)$
update $g_k = (g_k + \Delta g_k)$
increment $i$ by one
**od**

**Application of the Scheme in Beam Vibration Control** Knowing the correct sensor values and actuator forces at each location, we can compute the Lyapunov derivative for the beam by using Eq. 29. The general fault-balancer scheme described above is then adapted for the beam vibration control system as done in the adaptive fault-tolerant scheme.

**Theorem 5.** *The five node distributed control system for beam vibration resulting from the dynamic balancer fault-tolerant scheme is self-stabilizing fault-tolerant.*

*Proof.* We know from the 5 nodes on-off distributed control system tolerates one byzantine fault. Hence, in the dynamic balancer system, when the gain factor becomes sufficiently high so as to get the control forces to the maximum, the system is guaranteed to stabilize. □

In Fig. 7, we show the energy in the beam plotted against time (in seconds) for the adaptive dynamic scheme. As with the previous schemes, three cases are considered. The dynamic balancer scheme precisely balances the erroneous control value and recovers from the fault. The scheme relies on immediate feedback of the applied control forces and results in fastest convergence among the 3 schemes: dynamic, dynamic adaptive and dynamic balancer.

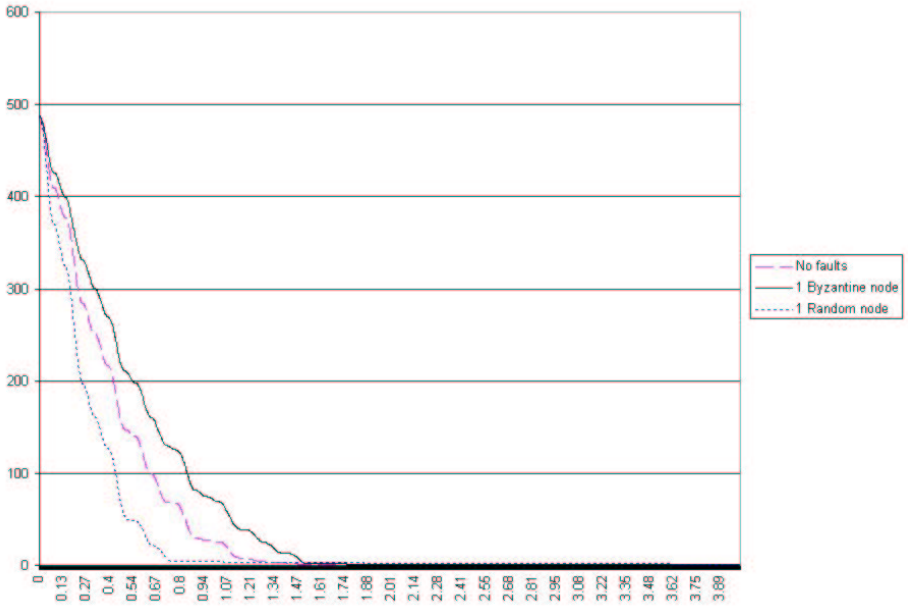


**Fig. 7.** Dynamic Balancer Fault-Tolerant Scheme (energy vs time )

## 4 Energy Spent in Control

In this section, we compare the energy spent to asymptotically stabilize the beam in the four schemes discussed above. In the static scheme, each node always applies a constant force, irrespective of the current sensed force. Hence it results in maximum energy spent to asymptotically stabilize the beam. The three versions of the linear scheme result in lesser energy spent in control. The graphs in Fig. 8 show the energy spent in control of the beam, corresponding to the experiments described in Section 3. In these experiments, the control level for the on-off scheme was set to 5. For each of the linear control schemes, the actuator forces were bounded by a maximum value of 5, thus making the comparisons meaningful. Also note that the random and byzantine disturbances are continuous and continue to persist even when the energy in the beam have brought down to a very small value. The energy spent in control shown in the graphs, is the energy required to bring down the energy in the beam to 1% of its original energy.

## 5 Conclusions

In this paper, based on Lyapunov functions, we formulated two types of fault-tolerant control for linear control systems: masking and self-stabilizing. Masking control systems retain their asymptotic stability in the presence of faults. We devised a static fault-tolerant scheme that yielded masking control, and demonstrated the scheme in the context of a beam vibration control system. In this

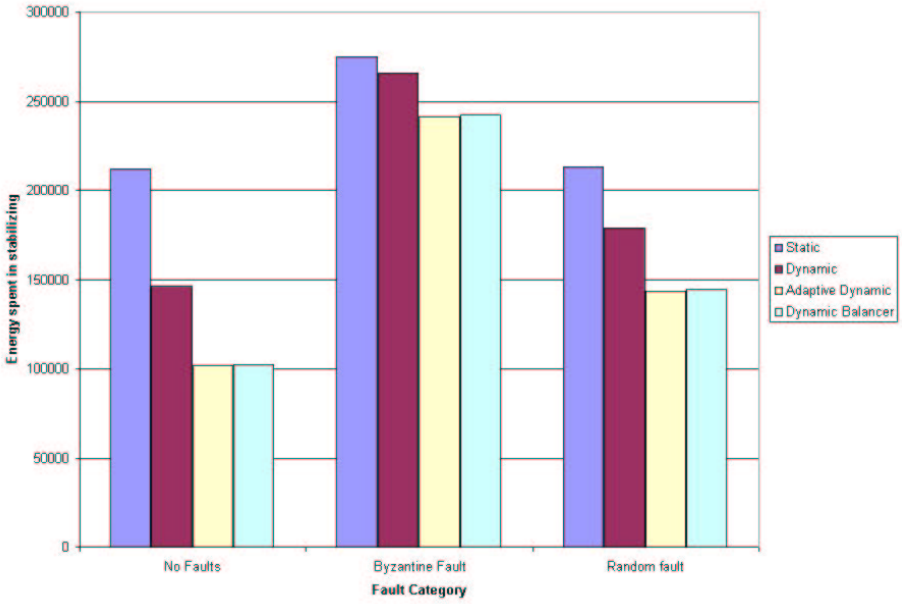


**Fig. 8.** Energy Spent to Asymptotically Stabilize the Beam

scheme, the control was made on-off at the maximum level, which meant that more energy could be spent than what was required if a byzantine node did not exert the worst possible forces. This constraint was relaxed in three successively more efficient schemes that yielded self-stabilizing control, which guaranteed eventual asymptotic stability of the control system. All schemes considered in the paper were distributed. Further, we did not assume the services of an underlying layer to detect the faults. In fact, the faults at the component level were abstracted to the control level and they were tolerated without any knowledge about their cause or nature.

An interesting direction for future research is to design systems that tolerate faults introduced by the middleware services due to unreliable communication channels, e.g. delays and omissions of messages. Regarding further extensions to our work, we would like to focus on fault-tolerant control theory for non linear and hybrid systems. We would also like to study the effect of continuous external perturbations on fault-tolerant control systems.

## References

1. J. H. Wensley, L. Lamport, J. Goldberg, M. W. Green, K. M. Lewitt, P. M. Melliar-Smith, R. E. Shostak, and C. B. Weinstock. SIFT: design and analysis of a fault-tolerant computer for air trac control. *IEEE*, 66(10):1240–1255, 1978.
2. G. Karsai, G. Biswas, T. Pasternak, and S. Narasimhan. Fault-adaptive control: a cbs application. In *8th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, April 2001.

3. C. Bonivento, A. Paoli, and L. Marconi. Fault-tolerant control for a ship propulsion system. In *European Control Conference*, 2001.
4. X. Koutsoukos, F. Zhao, H. Haussecker, J. Reich, and P Cheung. Fault modelling for monitoring and diagnosis of sensor-rich hybrid systems. In *Proc. IEEE Conference on Decision and Control*, Orlando Fl, 2001.
5. R. J. Patton, P. Frank, and R. Clark. *Fault Diagnosis in Dynamic Systems: Theory and Applications*. Prentice Hall, 1989.
6. M. Blanke, R. I. Zamanabadi, and Bogh. Fault tolerant control systems, a holistic view. *Control Engineering Practice*, 5(5), 1997.
7. S. S. Ge, Z. Sun, and T. H. Lee. Reachability and controllability of switched linear systems. In *American Control Conference*, 2001.
8. D. Liberzon and A. S. Morse. Basic problems in stability and design of switched systems. *IEEE Control Systems Magazine*, 19(5):59–70, 1999.
9. O. L. V. Costa, E. Filho, E. O. A. Boukas, and R. P. Marques. Constrained quadratic control of discrete-time markovian jump linear systems. *Automatica*, 35(4):617–629, 1999.
10. D. Šiljak. *Decentralized Control of Complex Systems*. Academic Press, 1991.
11. K. Frampton. A comparison of hierarchies for decentralized vibration control. *submitted to the Journal of Vibration and Acoustics*, 2001.
12. T. Hogg and B. A. Huberman. Controlling smart matter. *Journal of Smart Material Systems and Structures*, 7:R1–R14, 1988.
13. S. R. Hall, E. F. Crawley, and J. P. How. Hierarchic control architecture for intelligent structures. *Journal of Guidance, Control and Dynamics*, 14(3):503–512, 1991.
14. J. P. How. Local control design methodologies for a hierarchic control architecture. Master's thesis, Dept. of Aeronautics and Astronautics, MIT, Cambridge, 1990.
15. J. P. How. Local control design methodologies for a hierarchic control architecture. *Journal of Guidance, Control and Dynamics*, 15(3):654–663, 1992.
16. L. M. Silverberg. Uniform damping control of spacecraft. *Journal of Guidance, Control, and Dynamics*, 9(2):221–227, 1986.
17. L. Meirovitch. *Dynamics and Control of Structures*. John Wiley and Sons, 1990.
18. L. Meirovitch. *Methods of Analytical Dynamics*. McGraw-Hill, 1970.
19. R. Bellman. *Introduction to Matrix Analysis*. McGraw Hill, 2 edition, 1970.
20. J. S. Bay. *Fundamentals of Linear State Space Systems*. McGraw-Hill, 1999.
21. M. J. Balas. Direct output feedback control of large space structures. *Journal of Astronautical Sciences*, 27(2):157–180, 1979.
22. M. J. Balas. Direct velocity feedback control of large space structures. *Journal of Guidance and Control*, 2(3):252–253, 1979.
23. L. Shenhar and L. Meirovitch. Minimum-fuel control of high-order systems. *Journal of Optimization Theory and Applications*, 48(3):469–491, 1986.